A Proposal for Deploying Hybrid Knowledge Bases: the ADOxx-to-GraphDB Interoperability Case

Dimitris Karagiannis,
University of Vienna, Austria,
Faculty of Computer Science,
Knowledge Engineering Research Group,
dk@dke.univie.ac.at

Robert Andrei Buchmann,
Babeş-Bolyai University, Romania,
Faculty of Economics and Business Administration,
Business Informatics Research Center,
robert.buchmann@econ.ubbcluj.ro

Abstract

Graph Database Management Systems brought data model abstractions closer to how humans are used to handle knowledge – i.e., driven by inferences across complex relationship networks rather than by encapsulating tuples under rigid schemata. Another discipline that commonly employs graph-like structures is diagrammatic Conceptual Modeling, where intuitive, graphical means of explicating knowledge are systematically studied and formalized.

Considering the common ground of graph databases, the paper proposes an integration of OWL ontologies with diagrammatic representations as enabled by the ADOxx metamodeling platform. The proposal is based on the RDF-semantics variant of OWL and leads to a particular type of hybrid knowledge bases hosted, for proof-of-concept purposes, by the GraphDB system due to its inferencing capabilities. The approach aims for complementarity and integration, providing agile diagrammatic means of creating semantic networks that are amenable to ontology-based reasoning.

1. Introduction

Recent Bloor reports [1] have stated that "[graph databases] is the fastest growing segment of the database market" due to their support for: (i) easily handling many-to-many relationships; (ii) representing machine-readable semantics (thus enabling reasoning); (iii) graph analytics. Certain database management systems aim to cover all these aspects – see GraphDB [2], which was recently adopted for Springer Nature's SciGraph platform [3].

The work at hand repurposes the benefits of graph databases towards the goal of managing knowledge derived from diagrammatic (model) representations. Consequently, of particular interest for this work is the ability of GraphDB to manage knowledge representations that are governed by OWL ontologies in compliance with the formal semantics of the Resource Description Framework (RDF) [4]. The contribution of the paper is a method and mechanism for ensuring interoperability between an agile modeling environment and GraphDB as an enabler for ontology-driven knowledge management systems.

Diagrammatic conceptual modeling provides means for externalizing knowledge - i.e., the non-embodied relational knowledge distinguished by [5]. Standard modeling languages have traditionally employed graph-like diagrams, where relations are expressed as arrows or other type of graphical connectors whose semantics are mapped on visual characteristics (type of line, type of arrowhead etc.). Many-to-many relationships and n-ary relationships, for which graph databases are well-suited, are common in such graphical representations. The interpretation of visual connectors as "relationships" implies that modeling languages are governed by a well-defined conceptualization. The strength of modeling languages lies in the knowledge structures that emerge from them - i.e., the use of modeling software is not limited to graphical documentation; it may also aim for the creation of diagrammatic knowledge in support of various knowledge processes or systems. In the evolution of diagrammatic modeling, we are at a point where the initial goals of supporting human-to-human communication through graphical means can be complemented by a rich user experience (not limited to drawing static 2D shapes) as well as semantic interoperability (the focus of this paper).

Depending on the preferred viewpoint, a diagrammatic model may be perceived either (i) as a visual representation created to convey meaning, or (ii) as a semantic structure that has a graphical manifestation (semiotically-driven and possibly interactive). The work at hand favors the second viewpoint in order to bridge the practice of diagrammatic modeling with data management in

URI: http://hdl.handle.net/10125/50399
ISBN: 978-0-9981331-1-9

graph databases. GraphDB was selected to manage diagrammatic semantics due to its built-in reasoning mechanisms (OWL inference patterns and a custom rule engine).

Although application scenarios of the paper's proposal may be envisioned for standard modeling languages (e.g., UML, BPMN etc.), part of this contribution is to also stress the complementary benefit of the Agile Modeling Method Engineering (AMME) methodology [6] which allows a full customization of a modeling language in order to achieve alignment between the modeling semantics and the ontology running on GraphDB. Consequently, the paper's running example will not employ a standard modeling language; however, additional references will be provided to a standards-oriented implementation of the proposal.

Besides the diagrammatic contents and the (OWL-based) GraphDB ontologies, a third ingredient that may participate in the discussed hybridization is execution-time data. Recent versions of GraphDB include the OntoRefine plug-in (a migration of the formerly known Google Refine project [7]), which can import instance-level data from non-graph sources, off-line or on-line (e.g., Excel, CSV, JSON).

The three ingredients can form a hybrid knowledge base that is unified, from a representational perspective, by the Resource Description Framework, and, from a data management perspective, with the help of GraphDB.

The remainder of the paper is structured as follows: Section 2 will present the methodological and technological enablers, outlining the contribution context and overview. Section 3 will introduce a showcase modeling language as a running example, complemented by inference patterns based on the proposed interoperability mechanism. Section 4 generalizes the discussion beyond the showcase example. Section 5 comments on related works. The paper ends with a concluding section summarizing preliminary evaluations.

2. Methodology and Solution Summary

2.1. Agile Modeling Method Engineering

From a methodical point of view, the work at hand relies on the interplay between the Agile Modeling Method Engineering (AMME) [6] framework (its methodological aspect) and traditional ontology engineering methodologies.

The typical outcome of AMME is a modeling tool providing a fully customized diagrammatic language and related functionality. To achieve this, AMME provides a conceptualization method that has been crystallized by observing the development processes of numerous requirements-driven modeling methods and software – both for academic experimentation or targeted to industry (e.g., business process management products [8]). A large corpus of implementations has been contributed by methodologists within the collaborative environment of the Open Models Laboratory [9] (an overview of selected methods and their tools is published in [10]).

Fig. 1 indicates the general structure of one AMME iteration, as well as the focus of each stage – from initial knowledge acquisition to deployment and usage. The last stage is further detailed in terms of usage steps for the method introduced by this paper. The actual means of performing the steps are dependent on the underlying metamodeling platform on which the modeling tool is implemented (to be made visible later in platform-specific examples).

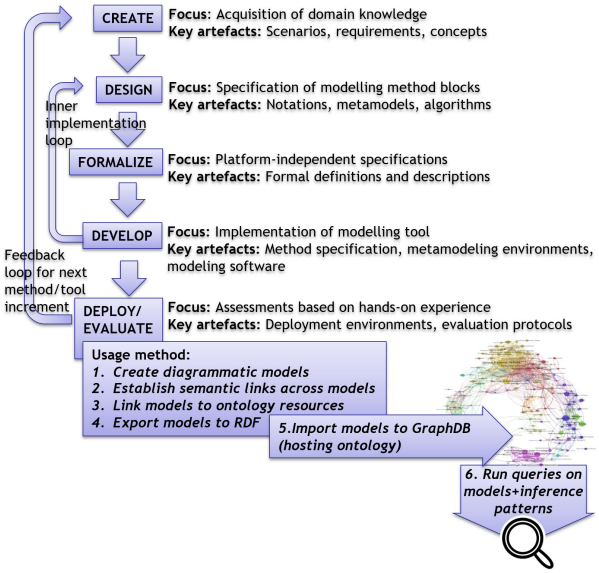

Figure 1. AMME: a methodological view

Since it deals with a conceptualization process, AMME shows some high-level similarities with traditional ontology engineering methodologies (catalogued in [11,12]). However, on a lower level, the conceptualization process is specialized with respect to the artefact's nature – here, a *modeling method and its implementation (modeling tool)*. Unlike an ontology, this places emphasis on notational dynamics (e.g., semantics manifesting on a graphical level or in the user experience) and model-based functionality (e.g., model-based reporting, process simulation, code generation).

The key artefact - the modeling method - is defined in terms of building blocks that can be agilely

customized for selected requirements. These building blocks are (cf. [13]) the *language* (modeling notation, syntax and semantics), the *modeling procedure* and the *model-based functionality* (mechanisms and algorithms). Together, they ensure that a modeling tool is not just an asemantic drawing tool, but that the knowledge expressed in a diagrammatic manner is also machine-readable (e.g., at least query-able) so that relevant functionality can be built of it. Such functionality is usually made available within the modeling tool – however, interoperability must also be considered to improve the value of models outside a modeling environment. In this paper's proposal, diagrammatic structures are exposed to GraphDB inferences (and further to external functionality that relies on them), since their graph-like nature makes them adequate for RDF-based semantics.

The entwinement between AMME and ontology engineering processes may be driven from either side, depending on where the modeling requirements originate: (a) the main purpose of the modeling language may be to support the extraction of knowledge to be subjected to the ontology - in this case the modeling requirements are subordinated to the ontology's competency questions; (b) the modeling language may have other purposes (e.g., simulation, model transformation) or may have existed in parallel with the ontology - in this case AMME acts as an alignment method.

2.2. ADOxx: fast prototyping support for AMME

ADOxx [14] is a metamodeling platform commonly employed as a fast prototyping environment for AMME (e.g., in projects developed by the Open Models Laboratory community [9]). It allows methodologists to incrementally develop their modeling method and to iteratively loop through the cycle suggested in Fig. 1. A meta-metamodel is built in ADOxx to allow the implementation of a modeling language notation, grammar and vocabulary, as well as the scripting of model-based functionality. The agility promoted by AMME manifests in several aspects enabled by ADOxx – *adaptability* (the ability to tailor the language for specific semantic or functional requirements), *extensibility* (the ability to extend existing languages), *integrability* (the ability to hybridize language fragments), *operability* (the ability to provide interaction mechanisms beyond the basic "diagram drawing"), *usability* (the ability to support model creation and comprehension through semantics-aware assistance features). A formalization of the ADOxx meta-metamodel is available in [15] and a selection of tools implemented on it was presented in [10].

At user interface level, the key model creation capabilities provided by any modeling tool implemented on ADOxx are suggested by a toy example ("cooking modeling language") in Fig. 2:

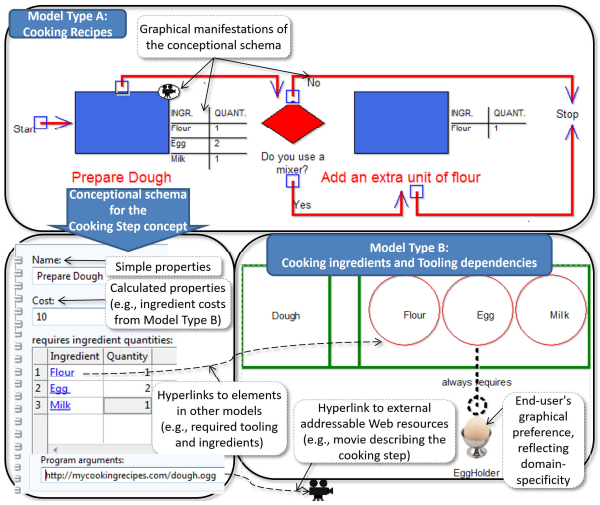

Figure 2. Various types of annotations in ADOxx-based modeling tools

The modeling language can be partitioned in multiple model types representing complementary viewpoints on the system to be modeled (in Fig. 2: "cooking recipes" and "cooking ingredients/tooling").

Diagrammatic elements are described by a "conceptional schema" that defines machine-readable properties to be exposed to the modeler for editing (in Fig. 2: the annotation sheet of the "Prepare Dough" cooking step); thus, diagrammatic semantics are not limited to human interpretation based on labeling or graphical meaning consensus. As Fig. 2 shows, annotation sheets support multiple ways of describing or linking model elements and are repurposed in the work at hand to derive RDF descriptions (including links to ontology terms or instance data available in GraphDB). Fig. 2 also shows that properties may dynamically and interactively manifest in notation – i.e., graphics are scripted (dependent on the annotations), thus providing richer semiotic opportunities compared to 2D static symbols.

Any of these aspects are subjected to agile customization through AMME, guided by "modeling requirements" (i.e., requirements on the semantic space and variability that the language should enable with respect to its intended use – here, alignment with OWL inference patterns on GraphDB).

In order to make diagrammatic contents available to OWL reasoning on GraphDB, an interoperability mechanism was implemented as an ADOxx plug-in

that generates RDF graphs from diagrammatic contents based on certain transformation patterns (a version of the plug-in is available at [16]). The transformation reads the internal ADOxx representation of models, strips away graphics and builds a collection of RDF graphs (in a user-provided namespace) containing descriptions of all model elements including their types, annotations, visual connectors, hyperlinks across models or between model elements and external resources (e.g., ontological terms or data). For example, in Fig. 2, the relation between the cooking step and its required ingredient (characterized by quantity) acts as a hyperlink in the modeling environment, but becomes a semantically rich relation in GraphDB, with resource identifiers (URIs) generated for all involved elements and properties. The OntoRefine plug-in of GraphDB may additionally enrich such knowledge structures with data that cannot be captured (or doesn't make sense to be captured) in the modeling environment – e.g., due to the fast changing nature of data.

3. Running Example

3.1. Showcase Modeling Language

As a base for inference pattern examples, Fig. 3 introduces a showcase modeling language, including a legend of its customized notation. The modeled scenario involves a virtual enterprise that provides make-to-order clothing, including courier services that are necessary along the production process.

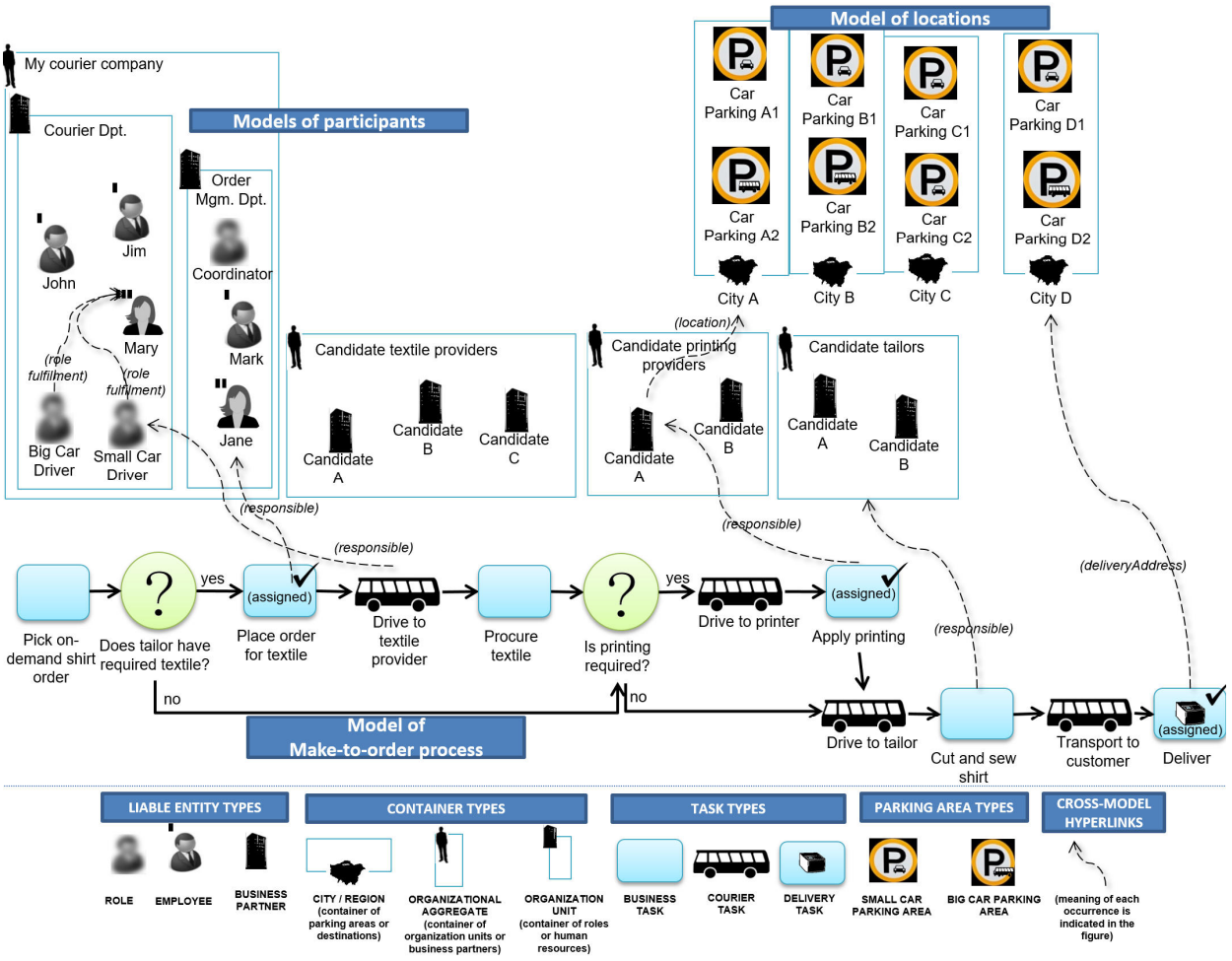

Figure 3. Model samples (including notation legend)

Domain-specificity manifests in notation, semantics and in partitioning the language over different types of models, making the example richer than what could be expressed with standard business process notations (the proposal is, however, also applicable to standards – see the generality considerations in Section 4).

Fig. 3 shows a make-to-order production-and-delivery process model extended (via the dotted

"hyperlinks") with several business context elements: (i) *models of the involved participants* – i.e., the coordinating company (including employees and courier roles), the candidate partners that can contribute to the various production process phases; (ii) *models of covered locations* – i.e., regions/cities and the parking areas that are available in each of them. Notice that not all relations manifest as visual connectors:

- some of them are visually indicated by relative position – i.e., a company contains departments, departments contain roles and employees; cities contain parking areas; candidate business partners are also grouped in containers based on their role/capability they can provide;
- other relations manifest as hyperlinks distinguished by machine-readable meaning: (i) a task may have attached a "responsible" which may be an instance (employee, concrete business partners) in an As-Is model; or a class (role, group of candidates) in a To-Be model; (ii) instance employees are linked to the roles they can fulfil (thus avoiding visual cluttering with additional connectors), business partners are linked to the cities where they are located, delivery tasks are linked to the city where the customer is etc.

In Fig. 4 a small fragment of this example is isolated to highlight the key transformation patterns (namespaces are avoided and URIs are adjusted for readability). The figure shows key elements of the schema governing the derived RDF graphs – parts of the schema are *dynamically derived from the language vocabulary* (as customized through AMME), while other parts are *fixed to distinguish between structural constituents* of the models (conforming the platform meta-metamodel, thus enabling transformation for any language implemented on it).

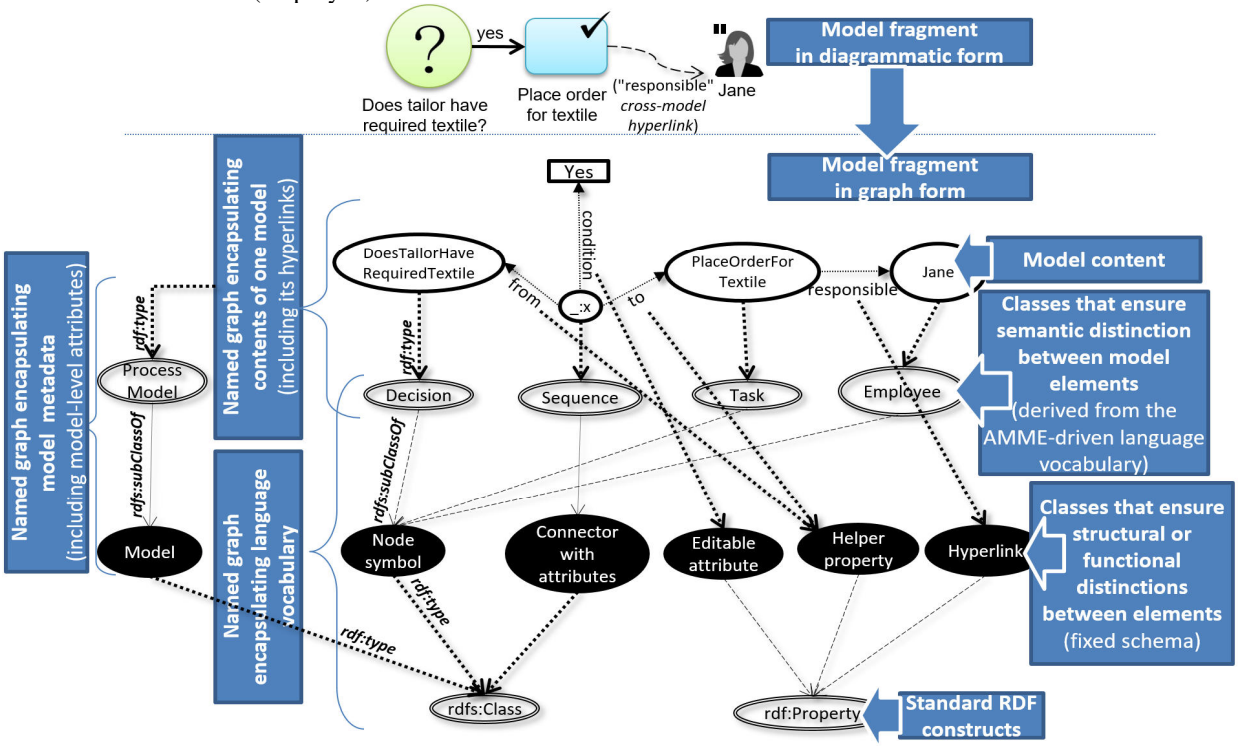

Figure 4. The core RDF graph constituents derived from diagrammatic content

A summary of transformation patterns is hereby provided, limited to the classes visible in Fig. 4 and to the inference patterns to be further analyzed (a more comprehensive discussion, including use cases not mentioned in this paper, is available in [17,18]): (a) each model becomes a distinct named graph (may be further described with model-level metadata); (b) each diagrammatic node becomes an RDF resource, described by the annotations defined in its conceptional schema; (c) visual connectors typically become RDF properties (derived from the language metamodel); (d) some visual connectors may have their own attributes, consequently they are treated as n-ary relations (e.g., the *x* node in Fig. 4); (e) hyperlinks also become RDF properties or n-ary relations, depending on their complexity; (f) a few fixed, "helper" properties are prescribed to support certain patterns (e.g., the *from/to* edges used to build the n-ary relation in Fig. 4; or, a *contains* relation between containers and their contents).

The next section presents several inference patterns that become possible once this structure is transferred to GraphDB. The beneficiaries of these inferences are arbitrary clients that query the graph server through its HTTP REST service.

3.2. Inference Patterns

Fig. 5 highlights the n-ary relationships that are derived from graphical connectors having their own conceptional schema. In this case the arrows showing the order the process tasks/decisions are annotated with transition conditions. The n-ary relation that links any two consecutive process steps makes the navigation via graph queries (i.e., SPARQL [19]) less efficient. OWL axioms contribute by enriching the initial graph with additional properties – e.g., a direct and transitive *followedBy* property will enable facile navigation along the process, whenever the connectors' data properties are not relevant.

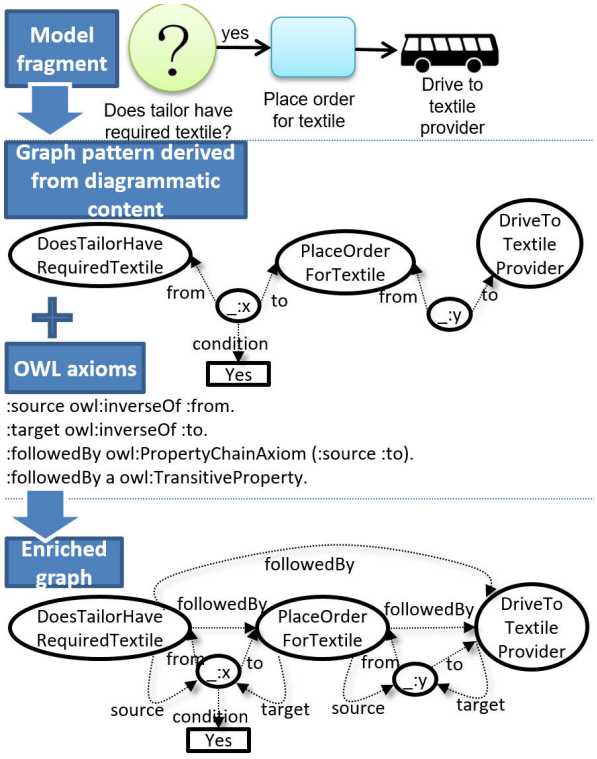

Figure 5. Property enrichment inferences

Fig. 6 highlights another inference pattern based on OWL axioms that enrich the typing of model elements beyond what is prescribed by the modeling language (the same visual coding as in Fig. 4 is used to depict hyperlinks, rdf:type and rdfs:subClassOf predicates).

An aspect that was not made explicit before is that the different types of tasks in the example (business task, courier/transportation task and delivery task) are not different concepts in the modeling language – it's a single concept whose graphics are scripted to show different icons based on certain property values (or hyperlink targets).

This is a notational feature of ADOxx that may be considered a "notational inference" – however, in GraphDB the graphic distinction must also become a semantic one. The axioms in Fig. 6 achieve this – i.e., the department of the responsible role determines whether a task is a "courier task".

A second way of enriching the task types is shown in the same figure (bottom side), where a dedicated annotation slot is provided directly in the modeling tool, allowing the user to assign arbitrary RDF statements to any node element, including additional types (e.g., from an already-in-use ontology).

This possibility of freely assigning RDF triples to model elements also provides the opportunity of bridging the diagrammatic contents with any execution-time data that makes sense to be semantically linked to model elements. Exemplary cases of bridging model elements with data are highlighted in Fig. 7:

(1) *Starting from a courier task, to reach phone numbers or availability information* for employees that may fulfil the task (the example inference is written in the figure as a custom Horst rule, a syntax available in GraphDB complementary to OWL);

(2) *Starting from a task, to reach the candidate business partners' contact/availability data*;

(3) *Starting from a task, to reach available parking options* (e.g, for a parking reservation app) or to *generate execution trace data*, with properties that are not necessarily available in the modeling language, but rather written by a client application that has write access to the GraphDB endpoint where the model information was exposed. The execution-time data may be imported into GraphDB via its OntoRefine plug-in or written into GraphDB by client applications – see, in Fig. 7, the INSERT (SPARQL) query creating process execution traces that are linked, for future analysis, to the process model element and the instance customer order that triggered the execution.

Some instance-level data may be directly stored as annotations of model elements; in that case it should be relatively stable data (e.g., phone numbers of employees). For fast-changing or dynamically generated data, having it stored in models would be inefficient (but not impossible, considering the data acquisition capabilities of ADOxx).

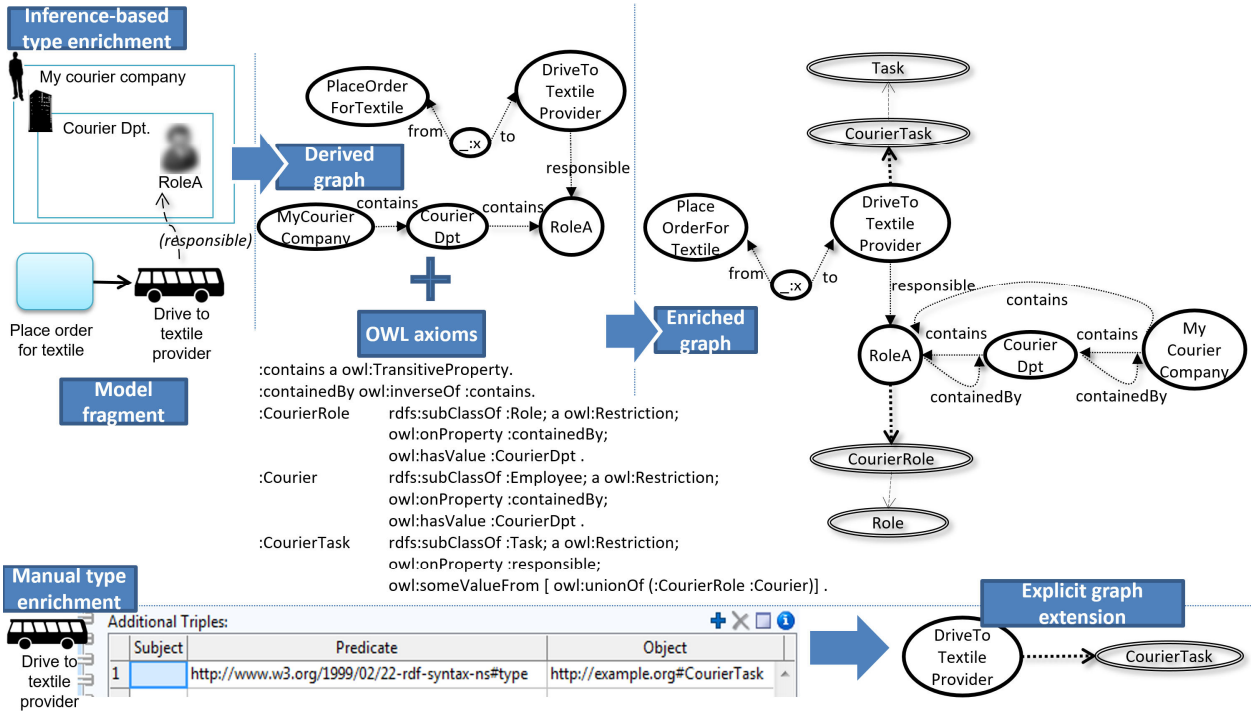

Figure 6. Type enrichment techniques for model elements

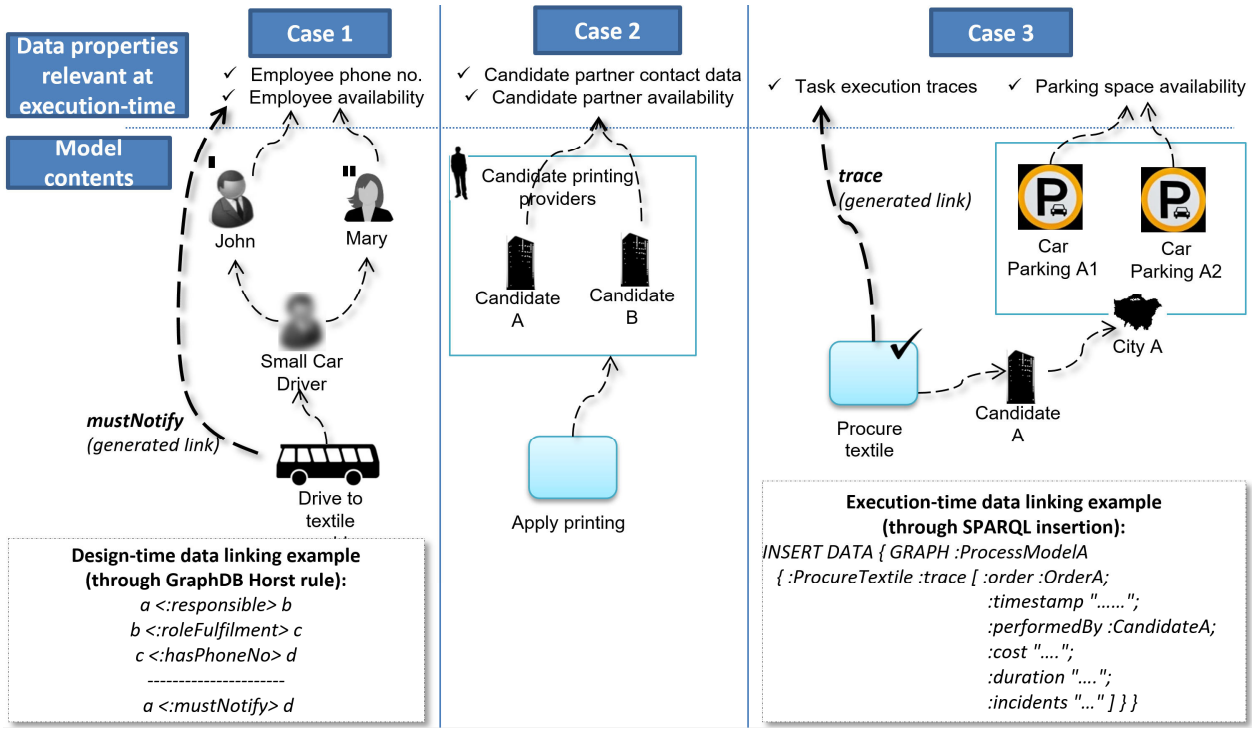

Figure 7. Bridging model contents with execution-time data

4. Generalization Discussion

The paper aims to decouple the proposed mechanism and method from a particular tool, language or standard, thus emphasizing the benefit of AMME for a full customization of the semantic space that is exposed to the proposed knowledge hybridization. Moreover, the mappings between

diagrammatic patterns and RDF graph structures discussed in [17, 18] may inspire implementations that are not based on ADOxx (e.g., based on the Meta-Object Facility [20]).

To emphasize that the proposal is similarly applicable to standard languages, a version of the model-to-RDF converter was made available in the BEE-UP modeling software [9] – a free tool published for educational purposes through the Open Models Laboratory portal. It supports modeling with well-known languages (UML, BPMN, ER, EPC and Petri Nets) plus extensions that have been agilely added to enrich the semantic space exposed by such languages – allowing, for example, the mapping of generic Petri Net transitions to activities described in organizational terms (e.g., responsibilities). The tool is targeted to users accustomed with modeling standards and is typically adopted in courses on business analysis, business process management or software design. The role of AMME was limited to the inclusion of a few semantic extensions in order to showcase cross-model reasoning.

5. Related Works

The paper promotes complementarity of heterogeneous knowledge resources – a "meet-in-the-middle" approach instead of the more traditional approaches of (i) converting between ontologies and models [21], (ii) stressing their distinctions and "essentially different roles" [22] or (iii) applying ontological evaluations on modeling languages [23, 24]. The T-box exported from diagrammatic models is quite minimal – sufficient to have a basic typing of diagrammatic elements that may be further linked to other T-boxes for type enrichment. Thus the proposal aims at enabling end-users to create linked knowledge structures with more intuitive means than the manual writing of graph serializations or the use of ontology editors.

The proposal may be considered a generalization of existing attempts to apply ontological reasoning on standard types of models (e.g., on UML [25], on business process models [26]). Such attempts take the modeling language as an invariant and, for proof-of-concept, employ Protege [27] or logic programming on XML model serializations. In the current proposal the modeling language is tailored in ADOxx and the hybrid knowledge structure is hosted and reasoned upon with the help of GraphDB to make it available to arbitrary client apps. Consequently, the proposal makes a strong case for Agile Modeling Method Engineering, beyond the methodology's original aim of enabling the deployment of domain-specific modeling languages; here, AMME is applied to mediate the alignment of language semantics with ontologies and execution-time instance data. Other methodologies enabling semantic customization of methods or languages (e.g., [28,29]) may also be repurposed towards this goal.

With a narrower scope, [30] proposes "semantic business process modeling" by enriching process models with process ontologies to compensate for the vagueness of natural language labels used in models. Other semantic annotation techniques, also targeting standard business process modeling languages have been collected in [31]. With AMME, the conceptional schema of modeling symbols defines machine-readable properties that are made available as "resource descriptions" (in RDF sense), hence labelling is not the only way to convey meaning. Also, the contribution is generalized beyond the scope of standard BPM languages (and at the same time, easily adaptable to those).

An overview of opportunities created by the proposed interoperability mechanism was previously published under the umbrella label of "Linked Open Models" [18], with several use cases based on semantic queries being detailed in [32].

6. Concluding Evaluation

The paper presented an approach to building hybrid knowledge bases, enabled by an ADOxx-to-GraphDB interoperability mechanism that employs the Resource Description Framework and the methodology of AMME to achieve an agile alignment between diagrammatic semantics and reasoning patterns within GraphDB.

Aiming for simplicity, a showcase modeling language was employed to make the argumentation easy to follow. Project-based validation with more complex languages is underway in the context of the EnterKnow project [33], whereas an early concept of the proposal, limited to SPARQL queries was initially investigated in earlier projects [32].

At this point only preliminary evaluations have been performed on students enrolled in semantic technology courses, where three means of knowledge graph creation were discussed and compared:

- Manual typing of RDF graphs in the user-friendliest serialization format, i.e. TriG;
- Creation of instance assertions in Protege;
- Diagrammatic modeling in an ADOxx-based modeling tool that supports this paper's proposal.

The assessment targeted the following metrics: (i) *learning effort* (the effort of learning how to use the tool / language / serialization syntax, measured by self-assessed percentage in the total learning time), (ii)

knowledge creation effort (the time taken to create the knowledge graphs corresponding to a given scenario), (iii) *knowledge quality* (assessed by the teacher evaluating the mistake rate in the final result).

Table 1 shows the averages for each assessment. Biases have been identified in the *learning effort* self-assessment, as the time allocated for learning was not necessarily the time required to learn the tool usage (post-assessment discussion revealed that it also included a preference factor). The other results recommend diagrammatic modeling as an intuitive way of constructing RDF knowledge graphs.

Table 1. Quantitative comparison of different knowledge creation means

	Diagrammatic modeling	Direct typing (TriG serialization)	Protege editing
Learning effort (percentage of total learning time)	39%	25%	36%
Knowledge creation effort (hours)	0,5	1,2	0,7
Knowledge graph quality (out of 10)	8,5	6,3	7,1

To conclude, a SWOT analysis summarizes in the following the current state of the paper's proposal:

Strengths: Diagrammatic conceptual models are graph-like data structures that provide an adequate use case for both (i) graph databases and (ii) agile customization of modeling tools/methods that deviate from standards for the benefits of domain-specificity or alignment to ontologies. Linking mechanisms can ensure semantic interoperability between diagrammatic elements, semantically lifted execution-data and OWL ontologies built on RDF semantics. The creation of RDF graphs through agile diagrammatic modeling methods may be more intuitive than using generic knowledge engineering means (i.e. ontology editors or knowledge graph serializations). The diagrammatic knowledge is treated here as a semantic complement to ontologies -i.e., the goal is hybridization and not replacement/generation of ontologies.

Weaknesses: Further evaluations are still necessary on industry-oriented languages or standard languages. Client-side proofs-of-concept are further necessary to bring the hybrid knowledge at end-user level, in "model-aware" front-ends.

Opportunities: Recent versions of GraphDB provide built-in support for geospatial ontologies (GeoSPARQL [34]), thus creating opportunities for geographical inferences, particularly relevant for the showcase language discussed in this paper.

The paper's proposal also creates opportunities for the field of Enterprise Modeling, where an enterprise is typically modeled in a multi-perspective manner [35]. The different perspectives may be expressed in different types of models (possibly created in different modeling environments), thus requiring semantic bridges that can benefit from GraphDB's reasoning mechanisms (e.g., for consistency checks).

Opponents of the Linked Data paradigm and RDF data model have been arguing that a "killer app" to justify the replacement of traditional data management technology has not emerged yet, extensive efforts being made to RDFize legacy data structures. The viewpoint that motivates the work at hand is that graph-like structures are close to the relationship-centric, intuitive way in which humans are used to externalize knowledge therefore the hereby advocated interoperability mechanism is a valuable integration opportunity for knowledge management systems.

Threats: The proposal depends on the uptake of graph databases with OWL reasoning support. Although the Bloor reports cited in the introduction is optimistic about graph databases in general, adoption of OWL and Semantic Web principles is still weakly represented in common applications.

Acknowledgments. The work of Dr. Robert Buchmann was supported by the Romanian National Research Authority through UEFISCDI, under grant agreement PN-III-P2-2.1-PED-2016-1140.

7. References

[1] P. Howard, "Graph Databases", http://www.bloorresearch.com/technology/graph-databases.

[2] Ontotext, GraphDB official website, http://graphdb.ontotext.com/.

[3] Springer, Springer Nature SciGraph – official website, http://www.springernature.com/gp/researchers/scigraph.

[4] W3C, The Resource Description Framework – reference webpage - http://www.w3.org/RDF/.

[5] W. D. Holford and P. Hadaya, "Addressing the tacit knowledge gap in knowledge systems across agential realism", Proceedings of HICSS 50, IEEE, 2017, pp. 4465-4474.

[6] D. Karagiannis, "Agile modeling method engineering", Proceedings of the 19th Panhellenic Conf. on Informatics, ACM, 2015, p 5-10.

[7] OntoRefine – the official project page, http://openrefine.org/

[8] BOC GmbH, ADONIS:CE – official website, http://en.adonis-community.com.

[9] Open Models Laboratory, The Bee-Up project page, http://austria.omilab.org/psm/content/bee-up/info.

[10] D. Karagiannis, H. C. Mayr, and J. Mylopoulos (eds.), Domain-specific Conceptual Modeling, Springer, 2016.

[11] N. Casellas, “Methodologies, Tools and Languages for Ontology Design,” Legal Ontology Engineering - Methodologies, Modeling Trends, and the Ontology of Professional Judicial Knowledge, Springer, 2011, pp. 57–109.

[12] O. Corcho, M. Fernandez-Lopez, and A. Gomez-Perez, "Methodologies, tools and languages for building ontologies. Where is their meeting point?", Data and Knowledge Engineering, 46(1), Elsevier, 2003, pp. 41-64

[13] D. Karagiannis and H. Kühn, "Metamodeling Platforms", Proceedings of EC-Web 2002 – DEXA 2002, LNCS 2455, Springer, 2002, p. 182.

[14] BOC GmbH, ADOxx metamodeling platform – official website, http://www.adoxx.org/live.

[15] H. G. Fill, T. Redmond, and D. Karagiannis, "Formalizing Meta Models with FDMM: the ADOxx Case", Proceedings of ICEIS 2012, LNBIP 141, Springer, 2012, pp. 429-451.

[16] Open Models Laboratory, ADOxx RDF plug-in, http://austria.omilab.org/psm/content/comvantage/downloadlist?view=downloads.

[17] R. A. Buchmann and D. Karagiannis, "Pattern-based Transformation of Diagrammatic Conceptual Models for Semantic Enrichment in the Web of Data", Procedia Computer Science 60, Elsevier, 2016, pp. 150-159.

[18] D. Karagiannis and R. A. Buchmann, "Linked Open Models: extending Linked Open Data with conceptual model information", Information Systems 56, Elsevier, 2016, 174-197.

[19] W3C, SPARQL 1.1 Query Language, http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[20] Object Management Group, Meta-Object Facility, http://www.omg.org/mof/.

[21] A. Pipitone and R. Pirrone, "A Hidden Markov Model for Automatic Generation of ER Diagrams from OWL Ontology", Proceedings of the 2014 IEEE Int. Conf. on Semantic Computing, IEEE, 2014, DOI 10.1109/ICSC.2014.19.

[22] F. Fonseca and J. Martin, "Learning The Differences Between Ontologies and Conceptual Schemas Through Ontology-Driven Information Systems", Journal of the Association for Information Systems 8(2), 2007, pp. 129-142.

[23] G. Guizzardi, "Ontological foundations for structural conceptual models", Doctoral dissertation, Enschede, The Netherlands, 2005.

[24] S. Anwar, "An Ontological Foundation for Agile Modeling with UML", Proceedings of AMCIS 2010, Assoc. for Information Systems, Paper 283.

[25] A. Panich and W. Vatanawood, "Detection of design patterns from class diagram and sequence diagrams using ontology", Proceedings of the 15th IEEE/ACIS Int. Conf. on Computer and Information Science, DOI 10.1109/ICIS.2016.7550771, 2016.

[26] C. Corea and P. Delfmann, "Detecting Compliance with Business Rules in Ontology-Based Process Modeling", Proceedings der 13. Internationalen Tagung Wirtschaftsinformatik, 2017, pp. 226-240.

[27] Stanford University, Protege – the official website, http://protege.stanford.edu/.

[28] U. Frank, "Domain-specific modeling languages: requirements analysis and design guidelines", Domain Engineering, Springer, 2013, pp. 133–157.

[29] S. Kelly, K. Lyytinen, and M. Rossi, "MetaEdit+ a fully configurable multi-user and multi-tool CASE and CAME environment", Seminal Contributions to Information Systems Engineering, Springer, 2013, pp. 109–129.

[30] O. Thomas and M. Fellmann, "Semantic Process Modeling – Design and Implementation of an Ontology-based Representation of Business Processes", Business & Information Systems Engineering 1(6), Springer, pp. 438-451.

[31] S. Smolnik, F. Teuteberg, and O. Thomas, Semantic Technologies for Business and Information Systems Engineering: Concepts and Applications, Business Science Reference, 2012.

[32] R. A. Buchmann and D. Karagiannis, "Enriching Linked Data with Semantics from Domain-specific Diagrammatic Models", Business and Information Systems Engineering 58(5), Springer, 2016, pp. 341-353.

[33] ***, EnterKnow project page, http://enterknow.granturi.ubbcluj.ro.

[34] OGC, The GeoSPARQL standard, http://www.opengeospatial.org/standards/geosparql.

[35] U. Frank, "Multi-Perspective Enterprise Modeling (MEMO) – Conceptual Framework and Modeling Languages", Proceedings of HICSS 35, IEEE, 2002, p. 1258-1267